

Administração de Sistemas

Sprint 2

Tópicos de Trabalho

Funcionalidades Sprint 2

- User Story 6.4.1
- User Story 6.4.2
- User Story 6.4.3
- User Story 6.4.4
- User Story 6.4.5
- User Story 6.4.6
- User Story 6.4.7
- User Story 6.4.8

[User Story 6.4.1]

[Us realizada por: Bruno Ribeiro - 1221352]

Como administrador do sistema quero que o deployment de um dos módulos do RFP numa VM do DEI seja sistemático, validando de forma agendada com o plano de testes.

[Procedimento]

Para a realização desta US, temos de iniciar uma máquina Debian 12 no ambiente do DEI e instalar os recursos necessários como por exemplo o nano, o dotnet, o lsof... Em seguida podemos criar o diretório onde vamos guardar os artefactos da aplicação e os seus logs e para isso usamos o comando “mkdir ~/apps/healthcare”. Agora temos de fazer a configuração do pipeline no GithubActions e para isso temos de criar Secrets que vão ter as seguintes informações:

Secrets

Variables

Repository secrets

New repository secret

Name	Last updated
SSH_KEY	2 days ago
VM_IP	2 days ago
VM_PORT	3 days ago
VM_USER	2 days ago

Os valores de VM_IP, VM_PORT, SSH_KEY e VM_USER são preenchidos de acordo com a nossa VM.

Agora criamos o script com o nome “deploy.yml” dentro do diretório “github/workflows/”.

```

.github > workflows >  deploy.yml
1  name: .NET Build, Test & Deploy
2
3  on:
4    schedule:
5      - cron: '0 6 * * 6'
6
7  jobs:
8    deploy:
9      runs-on: ubuntu-latest
10
11     steps:
12       - name: Checkout Repository
13         uses: actions/checkout@v3
14
15       - name: Transfer Backend to Remote Server
16         uses: appleboy/scp-action@v0.1.5
17         with:
18           host: {{ secrets.VM_IP }}
19           username: {{ secrets.VM_USER }}
20           key: {{ secrets.SSH_KEY }}
21           source: ./dddnetcore/*
22           target: ~/apps/healthcare
23
24       - name: Deploy Backend
25         uses: appleboy/ssh-action@v0.1.5
26         with:
27           host: {{ secrets.VM_IP }}
28           username: {{ secrets.VM_USER }}
29           key: {{ secrets.SSH_KEY }}
30           script: |
31             ~/apps/healthcare/deploy.sh
32
33

```

Explicação do script :

Utilizamos a chave “schedule” para definir a execução automática do deploy. O cronograma cron: '0 6 * * 6' garante que o deploy seja executado todos os sábados às 6:00 da manhã.

De seguida transferimos o código do backend do repositório para um servidor remoto (usando o SCP, um protocolo seguro de cópia de ficheiros entre máquinas), utilizando o host, o username e a ssh key para a conexão ser realizada com sucesso e o github ter acesso à máquina virtual. Definimos o src que é a pasta de onde os ficheiros vão ser copiados e um target que é o diretório na vm onde vão ser guardado os ficheiros.

Após os ficheiros terem sido copiados para o servidor remoto, a linha 35 executa um script de deploy (deploy.sh) para configurar e iniciar a aplicação no servidor.

Na máquina virtual através do comando “nano /apps/healthcare/deploy.sh” criamos o ficheiro com o seguinte código:

Explicação script:

```
GNU nano 7.2                                                                    deploy.
#!/bin/bash
# Parar o backend em execução na porta 5000

if lsof -i :5000 > /dev/null; then
    echo "Encerrando processo na porta 5000..."
    fuser -k 5000/tcp
else
    echo "Nenhum processo ativo na porta 5000."
fi

# Navegar até ao diretório da aplicação backend
cd ~/apps/healthcare/ddnetcore

# Instalar dependências
echo "A instalar dependências..."
dotnet restore

# Build do projeto
echo "Build do projeto a decorrer..."
dotnet build

# Testes do projeto
echo "Testes do projeto a correr..."
dotnet test

# Criar o diretório de logs se não existir
mkdir -p ~/apps/healthcare/logs

# Iniciar o servidor para a aplicação backend e mostrar a saída no terminal e no arquivo de log
echo "A iniciar o backend..."
nohup dotnet run -- --host 0.0.0.0 > ~/apps/healthcare/logs/backend.log 2>&1 < /dev/null &

echo "Deploy do backend realizado com sucesso"
```

O script começa por verificar se já existe algum processo a utilizar a porta TCP 5000. Se um processo estiver em execução na porta, ele será encerrado. Caso contrário, o script apenas informa que não há processo ativo nessa porta.

“cd ~/apps/healthcare/ddnetcore”, com este comando navegamos até à pasta onde está o projeto para depois podermos dar build (“dotnet build”) e testar (“dotnet test”) a aplicação.

“mkdir -p ~/apps/healthcare/logs”, cria caso não exista o diretório das logs(apenas vai fazer a primeira vez).

“nohup dotnet run -- --host 0.0.0.0 > ~/apps/healthcare/logs/backend.log 2>&1 < /dev/null &”, este comando inicia o servidor da aplicação backend. O comando “dotnet run” executa a aplicação .NET, e a opção --host 0.0.0.0 faz com que a aplicação aceite conexão em todas as interfaces de rede, tornando-a acessível fora do servidor. Através do “nohup” coloca a execução em segundo plano.

“~/apps/healthcare/logs/backend.log”, redireciona as mensagens de log para o ficheiro backend.log

Depois do ficheiro estar criado temos de dar permissões de execução ao utilizador root, neste caso. Para isso utilizamos o seguinte comando : “chmod +x ~/apps/healthcare/deploy.sh”.

Para testarmos a solução apresentado temos de retirar a linha referente ao agendamento do deployment.

← .NET Build, Test & Deploy

✓ Merge branch 'main' of https://github.com/Lapr5G21/sem5pi-24-25-3ddg21 #67 Re-run all jobs ...

Summary

Jobs

- ✓ deploy

Run details

Usage

Workflow file

deploy

succeeded now in 1m 32s

Search logs

- > ✓ Set up job 1s
- > ✓ Build appleboy/scp-action@v0.1.5 3s
- > ✓ Build appleboy/ssh-action@v0.1.5 2s
- > ✓ Checkout Repository 1s
- > ✓ Transfer Backend to Remote Server 12s
- > ✓ **Deploy Backend 1m 13s**
- > ✓ Post Checkout Repository 0s
- > ✓ Complete job 0s

Podemos também ver as logs na máquina virtual e verificar se o ficheiro ficou com os logs.

```

Terminal Shell Edição Visualização Janela Ajuda
brunao25 -- ssh -p 10350 root@vsgate-ssh.dei.isep.ipp.pt -- 204x55

Last login: Thu Nov 21 11:28:34 on ttys012
/dev/fd/12:18: command not found: compdef
brunao25@172-15-29-17 ~ % ssh -p 10350 root@vsgate-ssh.dei.isep.ipp.pt
^C
brunao25@172-15-29-17 ~ % ssh -p 10350 root@vsgate-ssh.dei.isep.ipp.pt
ssh: connect to host vsgate-ssh.dei.isep.ipp.pt port 10350: Operation timed out
brunao25@172-15-29-17 ~ % ssh -p 10350 root@vsgate-ssh.dei.isep.ipp.pt
root@vsgate-ssh.dei.isep.ipp.pt's password:
Linux vs350 4.15.0-213-generic #224-Ubuntu SMP Mon Jun 19 13:30:12 UTC 2023 x86_64
Debian GNU/Linux 12

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Nov 21 11:29:58 2024 from 10.9.0.38
root@vs350:~# cd apps
root@vs350:~/apps# cd healthcare/
root@vs350:~/apps/healthcare# cd logs/
root@vs350:~/apps/healthcare/logs# cat backend.log
info: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[62]
  User profile is available. Using '/root/.aspnet/DataProtection-Keys' as key repository; keys will not be encrypted at rest.
warn: Microsoft.AspNetCore.Server.Kestrel.Core.KestrelServer[1]
  The ASP.NET Core developer certificate is not trusted. For information about trusting the ASP.NET Core developer certificate, see https://aka.ms/aspnet/https-trust-dev-cert.
Hosting environment: Development
Content root path: /root/apps/healthcare/ddnetcore
Now listening on: https://localhost:5001
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
root@vs350:~/apps/healthcare/logs#

```

Como podemos ver o ficheiro ficou com as logs e está a aplicação a correr em segundo plano, com o auxílio do comando “ps aux | grep deploy” podemos ver que o processo está em segundo plano.

```

root@vs350:~/apps/healthcare/logs# ps aux | grep deploy
root    18687  0.0  0.1  6328  2060 pts/0    S+   14:34   0:00 grep deploy

```

Voltamos a colocar o scheduler no script do github e todas os sábados às 6 da manhã o servidor vai reiniciar e fazer quer o build quer os testes da aplicação.

[User Story 6.4.2]

[Us realizada por: João Oliveira - 1221957]

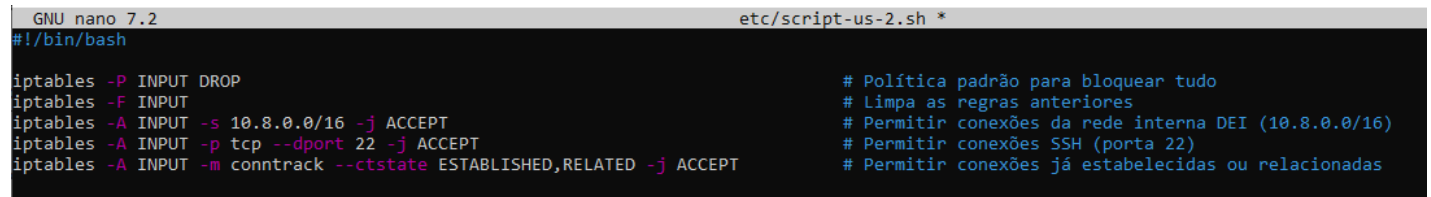
Como administrador do sistema quero que apenas os clientes da rede interna do DEI (cablada ou via VPN) possam aceder à solução.

[Procedimento]

Para restringir o acesso à solução apenas aos clientes da rede interna do DEI são necessárias algumas alterações na VM onde é feito deployment da nossa aplicação.

Para essa restrição teremos que limitar o acesso à gama de endereços do DEI. Observámos que a rede interna do DEI pertence a 10.8.0.0/16.

Foi desenvolvido o seguinte script para fazer a restrição pretendida usando o seguinte comando “**nano etc/script-us-2.sh**”:



```
GNU nano 7.2                               etc/script-us-2.sh *
#!/bin/bash

iptables -P INPUT DROP                    # Política padrão para bloquear tudo
iptables -F INPUT                        # Limpa as regras anteriores
iptables -A INPUT -s 10.8.0.0/16 -j ACCEPT # Permitir conexões da rede interna DEI (10.8.0.0/16)
iptables -A INPUT -p tcp --dport 22 -j ACCEPT # Permitir conexões SSH (porta 22)
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT # Permitir conexões já estabelecidas ou relacionadas
```

Figura 1 - script-us-2.sh

Em que cada linha faz o seguinte:

1. Define a política padrão da cadeia INPUT para *DROP*, ou seja, bloqueia todos os pacotes de entrada que não correspondam a regras específicas.
2. Limpa todas as regras existentes na cadeia INPUT.
3. Adiciona uma regra para permitir conexões de entrada provenientes da rede interna do DEI com a faixa de IPs 10.8.0.0/16.
4. Adiciona uma regra para permitir conexões de entrada no protocolo TCP direcionadas à porta 22, usada para SSH.
5. Permite pacotes de conexões que já estão estabelecidas ou relacionadas a conexões já iniciadas, possibilitando respostas da default gateway para garantir o funcionamento do SSH e a resolução de nomes DNS.

Para executar o script é necessário dar-lhe permissões de execução com o seguinte comando “**chmod +x script-us-2.sh**”, de seguida podemos então executar o script “**./script-us-2.sh**”.

Após executar este script, as regras serão aplicadas e só será possível aceder à solução através da rede interna do DEI.

Para que as alterações no iptables fiquem armazenadas mesmo após o reinício do sistema é necessário usar os seguintes comandos:

- **sudo apt install iptables-persistent netfilter-persistent -y** - para instalar o pacote netfilter-persistent.
- **sudo netfilter-persistent save** - para que as alterações ao iptables fiquem salvas.

[User Story 6.4.3]

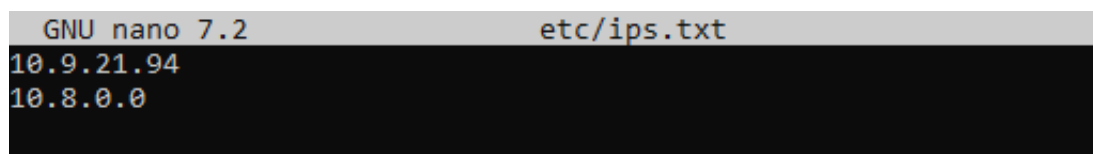
[Us realizada por: João Oliveira - 1221957]

Como administrador do sistema quero que os clientes indicados na user story 2 possam ser definidos pela simples alteração de um ficheiro de texto.

[Procedimento]

Esta user story pretende restringir o acesso à solução aos clientes da rede interna do DEI através de uma simples alteração de um ficheiro de texto. Para isso é necessário efetuar os seguintes passos na VM onde se encontra a solução:

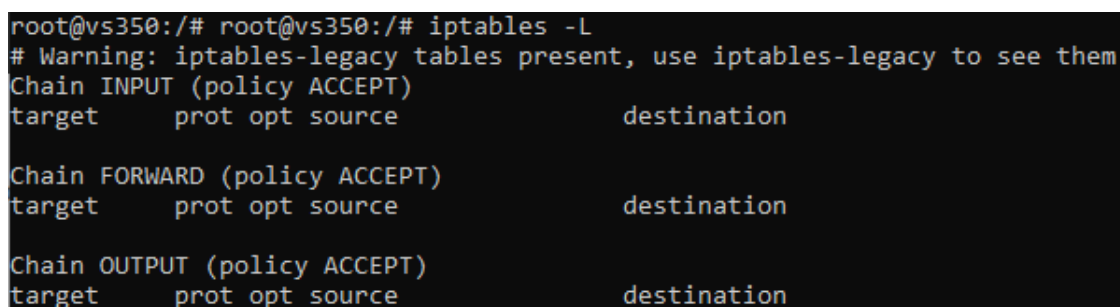
- Criar um ficheiro de texto onde estarão os IPs que vão ter acesso à solução usando o comando “**nano etc/ips.txt**”. Nesse ficheiro de texto deve ter o IP da máquina e o IP referente aos clientes da rede interna do DEI. O IP da máquina está na primeira linha e o IP referente aos clientes da rede interna do DEI está na segunda linha:



```
GNU nano 7.2      etc/ips.txt
10.9.21.94
10.8.0.0
```

Figura 2 - ficheiro de texto com os endereços IP

- Efetuar o reset à máquina virtual para não existir qualquer regra no filtro de pacotes IPv4.
- Executar o comando “**iptables -L**” para verificar as regras existentes no filtro de pacotes IPv4, não existe nenhuma como podemos ver na seguinte figura:



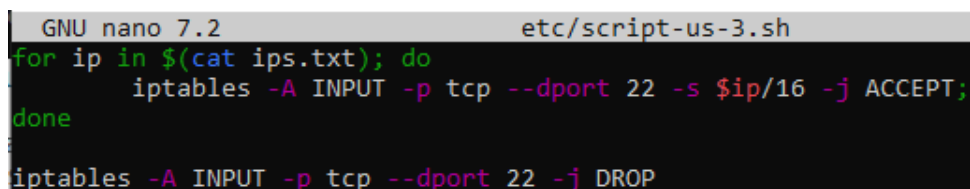
```
root@vs350:/# root@vs350:/# iptables -L
# Warning: iptables-legacy tables present, use iptables-legacy to see them
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

Figura 3 - regras do filtro de pacotes antes da realização da US

- Criar um script usando o comando “**nano etc/script-us-3.sh**”, que permite o acesso à solução apenas aos clientes que estão indicados no ficheiro de texto “**ips.txt**”. Este script contém um ciclo for que, para cada IP presente no ficheiro de texto, vai adicionar uma regra ao iptables que permite o tráfego TCP na porta 22 (SSH) para o intervalo de endereços IP especificado. Por fim ele contém também uma regra que irá bloquear todas as outras ligações. O script é o seguinte:



```
GNU nano 7.2      etc/script-us-3.sh
for ip in $(cat ips.txt); do
    iptables -A INPUT -p tcp --dport 22 -s $ip/16 -j ACCEPT;
done
iptables -A INPUT -p tcp --dport 22 -j DROP
```

Figura 4 - script-us-3.sh

- Após a criação do script é preciso dar-lhe permissão de execução usando o comando “**chmod +x script-us-3.sh**”.
- Agora corremos o script “**./script-us-3.sh**” e de seguida verificamos através do comando “**iptables -L**” que a restrição foi feita com sucesso:

```

root@vs350:/etc# chmod +x script-us-3.sh
root@vs350:/etc# ./script-us-3.sh
root@vs350:/etc# iptables -L
# Warning: iptables-legacy tables present, use iptables-legacy to see them
Chain INPUT (policy ACCEPT)
target     prot opt source                destination          tcp dpt:ssh
ACCEPT     tcp  --  10.9.0.0/16             anywhere             tcp dpt:ssh
ACCEPT     tcp  --  10.8.0.0/16             anywhere             tcp dpt:ssh
DROP       tcp  --  anywhere                anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination

```

Figura 5 - iptables após executar o script

- Por fim para que as mudanças que fizemos ao iptables fiquem guardadas mesmo após reiniciar o sistema fazemos o seguinte:
 - Primeiro instalamos pacote **netfilter-persistent** usando o comando “**sudo apt install iptables-persistent netfilter-persistent -y**”
 - De seguida, usamos o comando “**sudo netfilter-persistent save**” para que as alterações ao iptables fiquem salvas.

```

root@vs350:~# root@vs350:~# sudo netfilter-persistent save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
# Warning: iptables-legacy tables present, use iptables-legacy-save to see them
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save

```

Figura 6 - Alterações do iptables guardadas

[User Story 6.4.4]

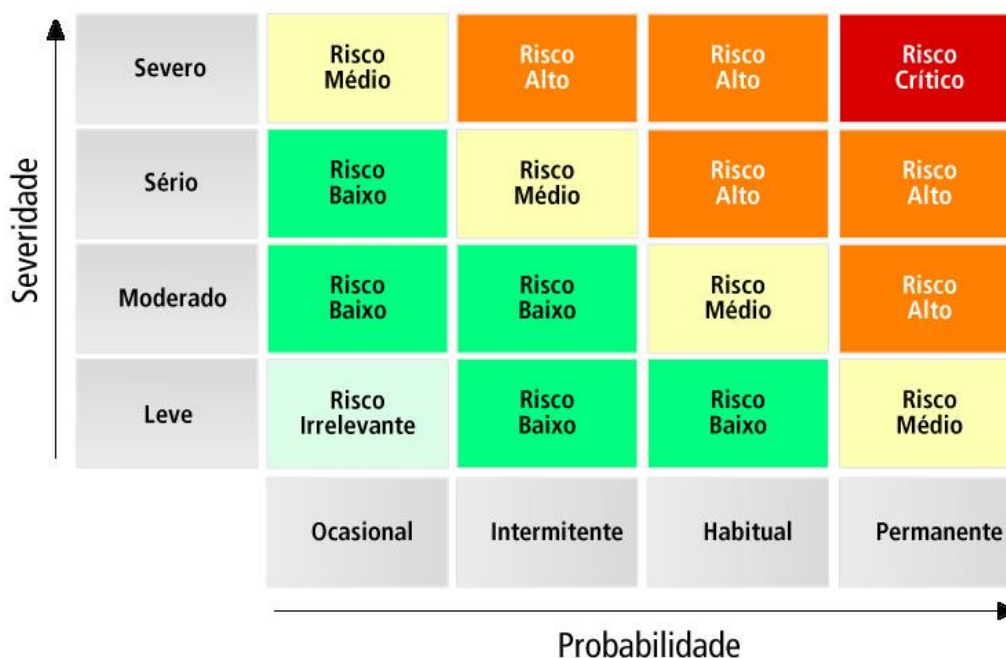
[Us realizada por: Eduardo Ferreira - 1230430]

Como administrador do sistema quero identificar e quantificar os riscos envolvidos na solução preconizada.

[Procedimento]

No requisito acima apresentado é pedida a avaliação de riscos e quantificação dos mesmos para a solução apresentada. A abordagem mais correta para essa avaliação deverá passar por uma matriz de riscos e identificação dos mesmos.

A Matriz permite a relação entre a probabilidade de um acontecimento e da severidade desse mesmo acontecimento, através da relação entre esses dois fatores é possível quantificar o risco.



Para o apuramento dos riscos associados á solução é necessária a definição dos acontecimentos possíveis e avaliação da sua severidade e probabilidade para resultar no seu risco:

Acontecimento	Probabilidade	Severidade	Risco Associado
Blackout do Sistema	Ocasional	Severo	Médio
Fuga de dados	Ocasional	Sério	Baixo
Sistema Lento	Habitual	Leve	Baixo
Autenticação Indevida	Intermitente	Severo	Alto
Erro humano	Habitual	Moderado	Médio

[User Story 6.4.5]

[Us realizada por: Vasco Teixeira - 1220839]

Como administrador do sistema quero que seja definido o MBCO (Minimum Business Continuity Objective) a propor aos stakeholders.

[Procedimento]

O MBCO (Minimum Business Continuity Objective) é o nível mínimo de serviços ou funcionalidades que devem ser garantidos durante uma interrupção para evitar que as operações da organização sejam gravemente comprometidas.

A missão que nos foi proposta foi configurar um sistema hospitalar para gerir as várias marcações de operações e os gerenciar os vários recursos, incluindo o registo de dados dos vários utilizadores, incluindo pacientes e staffs.

No projeto que estamos a desenvolver, o MBCO é definido como a capacidade do sistema de manter operações essenciais de agendamento de cirurgias e gestão de recursos durante interrupções. Isso implica que algumas funcionalidades principais devem continuar disponíveis, mesmo que de forma reduzida, para evitar a paralisação dos serviços.

Relativamente às funcionalidades essenciais a serem mantidas, é fundamental abordar os vários módulos que compõem o sistema. Começando pela interface que permite a interação entre os utilizadores e o núcleo da aplicação, a que designamos *front-end*, foi definido que o tempo em que pode estar em baixo é no máximo 1h30m, ou seja, concluído este tempo tem que ser permitido ao utilizador poder gerir os seus recursos de uma forma mais básica, sendo que após (no máximo) 14 horas já tem que estar este componente no seu estado normal.

Falando agora do núcleo do sistema, responsável por gerir e processar os pacientes, staffs e operações/cirurgias, *back-end*. Para este foi definido o mesmo tempo e as mesmas condicionantes do *front-end*, ou seja, no máximo 1h30m em que pode estar sem funcionar corretamente e no máximo 14 horas para a sua recuperação, pois consideramos um módulo de um nível bastante importante.

Abordando agora outro componente, o *3D Visualization Module*, que é responsável pela vista em tempo-real da ocupação das salas de cirurgias, para este existe uma prioridade menor em

relação aos outros dois já apresentados acima, isto é, foi determinado que após 5 horas do seu não funcionamento, este tem que apresentar um plano básico onde se pode observar de uma maneira muito simplista se estão a existir cirurgias ou não, sendo que após finalizadas 42 horas, este deve ser restaurado.

Depois existe ainda a parte que diz respeito à otimização e processamento do agendamento das operações, levando em conta a disponibilidade de salas, incluindo também os profissionais e prioridades médicas, o módulo *planning*. Aqui é definido um tempo de 4 horas em que o sistema pode permanecer sem este componente, onde após concluídas estas horas, este tem de apresentar um agendamento básico sem automatismos temporariamente. Posto isto, passado 24 horas desde a quebra do sistema, o estado tem que ser integralmente restabelecido.

Por último, o acesso aos dados em conformidade com o modelo RGPD também tem de ser garantido, permitindo que mesmo com todas estas situações, os dados de pacientes e staff possam ser acedidos de forma segura e conforme as regras do Regulamento Geral de Proteção de Dados.

Destacar que são também ativados mecanismos de comunicação interna (e-mails ou alertas) entre administradores, pacientes e staffs para informar problemas no sistema ou a necessidade de certos procedimentos manuais.

Com este MBCO, procuramos garantir que, mesmo em situações imprevistas, o sistema possa continuar a facilitar o agendamento de cirurgias, a segurança dos pacientes e a conformidade com obrigações legais.

[User Story 6.4.6]

Como administrador do sistema quero que seja proposta, justificada e implementada uma estratégia de cópia de segurança que minimize o RPO (*Recovery Point Objective*) e o WRT (*Work Recovery Time*).

[Us realizada por: Eduardo Ferreira - 1230430]

Na user storie acima descrita é pedida a definição de estratégias de cópia de segurança para a minimização do RPO (*Recovery Point Objective*) onde é definida a máxima quantidade de informação de admissível perda em caso de falha no sistema e também o WRT (*Work Recovery Time*) que é o tempo para restabelecer o sistema após uma falha critica.

A estratégia de copia de segurança proposta para este sistema consistirá em dois tipos de cópias de segurança, todos os dias exceto sábado será efetuada uma copia de segurança às 00:00 do tipo incremental e no dia proposto com menos fluxo, sábado, será executada uma copia total da base de dados também as 23:50.

Com a solução proposta acima estamos a definir o RPO para o fluxo de informação de 24 h. Os scripts desenvolvidos abaixo são a definição das copias de segurança totais e incrementais, com estes scripts e um agendamento adequado será garantida a integridade dos dados a cada 24H.

```
GNU nano 5.4 backup_dataBase_total.sh
#!/bin/bash

USER="root"
PASSWORD="DL8eW3HTyiQ3"
DB="vs1379.dei.isep.ipp.pt"
DIR_BACKUP="/backup_scripts/database/$(date '+%Y-%W')"
DATE=$(date '+%Y-%m-%d_%H-%M-%S')

mkdir -p $DIR_BACKUP

mysqldump -u $USER -p$PASSWORD $DB > $DIR_BACKUP/backup.$DATE.total.sql

tar -czf $DIR_BACKUP/backup.$DATE.total.tar.gz -C $DIR_BACKUP backup.$DATE.total.sql

rm $DIR_BACKUP/backup.$DATE.total.sql
```

```
GNU nano 5.4 backup_dataBase_incremental.sh
#!/bin/bash

USER="root"
PASSWORD="DL8eW3HTyiQ3"
DB_NAME="vs1379.dei.isep.ipp.pt"
DIR_BACKUP="/backup_scripts/database/$(date '+%Y-%W')"
DATE=$(date '+%Y-%m-%d_%H-%M-%S')

mkdir -p $DIR_BACKUP

mysql .u $USER -p$PASSWORD -D $DB_NAME -e "$QUERY" > $BACKUP_DIR/backup.$DATE.incremental.sql

tar -czf $DIR_BACKUP/backup.$DATE.incremental.tar.gz -C $DIR_BACKUP backup.$DATE.incremental.sql

rm $BACKUP_DIR/backup.$DATE.incremental.sql
```

No caso de uma falha será necessária uma restauração dos dados, para isso foi criado o ficheiro `restoration_scripts.sh` onde irá restaurar os dados a partir do último backup incremental caso este exista, senão utiliza o backup total. O WRT estará diretamente relacionado com o volume de dados entre cada copia, ou no caso de uma perda total, de todos os dados previamente existentes no sistema.

```

GNU nano 5.4                                restoration_scripts.sh
#!/bin/bash

dir_backup="/backup/database/$(date '%Y-%V')"

destination="/restores"

total_backup = ($ls ${dir_backup}/backup.*.total.tar)
tar -xf $total_backup -p -C ${destination}

incremental_backup=$(ls ${dir_backup}/backup.*.increment.tar 2> /dev/null)

if [ -z "${incremental_backup}" ]
then
echo "No incremental backups found"
else
do
    for backup in $incremental_backup
do
        tar -xf ${backup} -p -G -C ${destination}
done
fi

mv "${destination}/dump" "${destination}/$(date '+%Y-%m-%d')_restoration"

echo "Backup restoration complete"

exit 0

```

Para a definição da periodicidade das cópias foi editado o ficheiro crontab para que tal como já descrito execute a copia total no sábado às 00:00 e nos restantes dias a cópia periódica.

```

GNU nano 5.4                                crontab *
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
0 0 * * 0-5 root /backup_scripts/backup_dataBase_incremental.sh
0 0 * * 6 root /backup_scripts/backup_dataBase_total.sh_
#

```

[User Story 6.4.7]

[Us realizada por: Vasco Teixeira - 1220839]

Como administrador do sistema quero definir uma pasta pública para todos os utilizadores registados no sistema, onde podem ler tudo o que lá for colocado.

[Procedimento]

No desenvolvimento desta funcionalidade pretende-se definir uma pasta pública para todos os utilizadores registados no sistema (excluindo os administradores) para poder incluir documentos, etc com exclusividade de leitura que possam ser necessários. Para a realização da mesma, é condição inicial estar com a sessão iniciada como root na máquina virtual. Feito isto, passa-se então à execução dos devidos passos.

Primeiramente é necessário criar o diretório através do seguinte comando:

```
root@uvm021:/# mkdir /public_folder
```

Concluído este passo, sucede-se a atribuição de permissões de leitura na pasta criada para todos os utilizadores registados no sistema:

```
root@uvm021:/# chmod -R a+r /public_folder/
```

Destacar que neste comando o “-R” significa que esta alteração será aplicada a todos os arquivos e subdiretórios dentro desta pasta já ou futuramente criados. O “a” representa todos e o “+r” a adição de permissões de leitura.

De seguida, é necessário criar um ficheiro para testar as permissões definidas, o que pode ser feito a partir do seguinte comando:

```
root@uvm021:/# nano /public_folder/teste.txt
```

Agora é colocar algum texto no ficheiro e guardar:

```
GNU nano 5.4 /public_folder/teste.txt * M
Teste para verificar as permissões (só) de leitura como pretendido.
Asist grupo 21
```

Após estas etapas é verificar as permissões que se encontram neste ficheiro, o que pode ser feito através do comando “ls -l/public_folder”. Como é possível visualizar na imagem abaixo, o ficheiro teste.txt tem como permissões “-rw-r--r--” o que significa que o proprietário tem permissões de escrita e leitura (rw) e o grupo e outros utilizadores permissões de leitura (r):

```
root@uvm021:/# ls -l /public_folder/
total 4
-rw-r--r-- 1 root root 96 Nov 22 12:35 teste.txt
```

Agora o próximo e último passo é testar o acesso de um utilizador à pasta via ssh incluindo as respetivas permissões. Neste caso utilizou-se o utilizador luser99 para entrar na pasta partilhada (public_folder) e após isso realizou-se uma leitura ao conteúdo do ficheiro teste.txt, o que foi realizado com sucesso:

```
luser99@uvm021:~$ cd /public_folder
luser99@uvm021:/public_folder$ cat teste.txt
Teste para verificar as permissões (só) de leitura como pretendido.
Asist grupo 21
```

Na imagem abaixo, é possível observar a tentativa do utilizador em editar o conteúdo do ficheiro teste.txt. Como apenas possui permissões de leitura é expectável o insucesso da operação como se pode verificar através daquela linha a vermelho ao fundo da figura:

```
GNU nano 5.4 teste.txt *
Teste para verificar as permissões (só) de leitura como pretendido.
Asist grupo 21
```

[Error writing teste.txt: Permission denied]

```
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo
```

Como última testagem, é possível verificar a tentativa em criar um ficheiro na pasta partilhada, e como tal é expectável o insucesso da operação devido à inexistência de permissões de escrita em “public_folder”:

```
luser99@uvm021:/public_folder$ "Tentativa erro" > tentativa.txt  
-bash: tentativa.txt: Permission denied
```

[User Story 6.4.8]

[Us realizada por: Bruno Ribeiro - 1221352]

Como administrador do sistema quero obter os utilizadores com mais do que 3 tentativas de acesso incorretas.

Para a realização desta user story começamos por criar o ficheiro fail_login_users.sh com o comando “nano fail_login_users.sh”:


```

GNU nano 5.4                                fail_login_users.sh
#!/bin/bash

#limit of failed logins
LIMIT=$1

#output file
output_file="${LIMIT}_or_more_failed_login_users_$(date +%Y-%m-%dT%H:%M:%S.%3NZ').txt"

sudo grep "Failed password" /var/log/auth.log | awk '{print $9}' | sort | uniq -c | \
awk -v limit="$LIMIT" '{if ($1 > limit) print $2 " teve " $1 " logins errados."}' > "$output_file"

echo "Users with more than $LIMIT logins failed saved in the file"

```

Explicação do script:

Começamos por definir a variável LIMIT com o valor do primeiro argumento passado ao executar o script (\$1). Este valor determina o número mínimo de tentativas falhadas de login para incluir um utilizador no output. Criamos o nome do ficheiro de saída com o seguinte formato “N_or_more_failed_login_users_YYYY:mm:DD:HH::MM:SS:sssZ.txt”. Depois com o comando “grep “Failed password” /var/log/auth.log | awk “{print \$9} | sort | uniq -C” usamos o comando grep para procurar entradas no ficheiro /var/log/auth.log que contenham a mensagem “Failed password”, com o “awk “{print \$9}”” seleccionamos a 9ª coluna de cada linha, pois essa coluna contém o nome do utilizador que tentou fazer login. Com o “sort” e o “uniq -C” ordenamos alfabeticamente os utilizadores e conta o número de vezes que aparece cada um deles no ficheiro auth.log. “awk -v limit="\$LIMIT" '{if (\$1 >= limit) print \$2 " teve " \$1 " logins errados."}' > "\$output_file", com esta linha apenas fazemos a comparação entre o \$1(número de falhas de um user) com o valor limite e caso seja superior escrever no ficheiro “output_file” os logins errados daquele user.

Depois de guardado o script temos de dar permissões de execução com o comando “chmod 700 fail_logins_users.sh”.

Para testarmos esta funcionalidade apenas temos de executar o script e analisar o output.

```

root@uvm021:/etc# ./fail_login_users.sh 3
Users with more than 3 logins failed saved in the file
root@uvm021:/etc# cat 3_or_more_failed_login_users_2024-11-23T17\:\03\:\38.734Z.txt
; teve 13 logins errados.
luser99 teve 6 logins errados.
root teve 26 logins errados.

```

Neste caso definimos o limite como 3, como era pedido na user story e podemos analisar os users que

têm mais do que 3 logins. Podemos também realizar outro teste para comprovar que está a ir buscar realmente os users de acordo com o limite definido.

```
root@uvm021:/etc# ./fail_login_users.sh 14
Users with more than 14 logins failed saved in the file
root@uvm021:/etc# cat 14_or_more_failed_login_users_2024-11-23T17\.:05\:57.2372.txt
root teve 26 logins errados.
```

Como podemos ver agora definimos o limite como 14 e apenas o utilizador root aparece pois era o único com o número de logins errados acima de 14.